1997

A Morph-Based Simulated Annealing Heuristic for a Modified Bin-Packing Problem

Michael J. Brusco
Florida State University

Gary Thompson
Cornell University School of Hotel Administration, gmt1@cornell.edu

Larry W. Jacobs
Northern Illinois University

A Morph-Based Simulated Annealing Heuristic for a Modified Bin-Packing Problem

Abstract

This paper presents a local-search heuristic, based on the simulated annealing (SA) algorithm for a modified bin- packing problem (MBPP). The objective of the MBPP is to assign items of various sizes to a fixed number of bins, such that the sum-of-squared deviation (across all bins) from the target bin workload is minimized. This problem has a number of practical applications which include the assignment of computer jobs to processors, the assignment of projects to work teams, and infinite loading machine scheduling problems. The SA-based heuristic we developed uses a morph-based search procedure when looking for better allocations. In a large computational study we evaluated 12 versions of this new heuristic, as well as two versions of a previously published SA-based heuristic that used a completely random search. The primary performance measure for this evaluation was the mean percent above the best known objective value (MPABKOV). Since the MPABKOV associated with the best version of the random-search SA heuristic was more than 290 times larger than that of the best version of the morph-based SA heuristic, we conclude that the morphing process is a significant enhancement to SA algorithms for these problems.

Keywords

bin packing, heuristics, simulated annealing

Disciplines

Operations and Supply Chain Management

Comments

A MORPH-BASED SIMULATED ANNEALING HEURISTIC FOR A MODIFIED BIN-PACKING PROBLEM

Michael. J. Brusco

Florida State University

Gary M. Thompson

Cornell University

Larry W. Jacobs

Northern Illinois University

Correspondence to:
MJ Brusco
Information and Management Science Department
College of Business
Florida State University
Tallahassee, FL 32306-1042

Abstract

This paper presents a local-search heuristic, based on the simulated annealing (SA) algorithm for a modified bin- packing problem (MBPP). The objective of the MBPP is to assign items of various sizes to a fixed number of bins, such that the sum-of-squared deviation (across all bins) from the target bin workload is minimized. This problem has a number of practical applications which include the assignment of computer jobs to processors, the assignment of projects to work teams, and infinite loading machine scheduling problems. The SA-based heuristic we developed uses a morph-based search procedure when looking for better allocations. In a large computational study we evaluated 12 versions of this new heuristic, as well as two versions of a previously published SA-based heuristic that used a completely random search. The primary performance measure for this evaluation was the mean percent above the best known objective value (MPABKOV). Since the MPABKOV associated with the best version of the random-search SA heuristic was more than 290 times larger than that of the best version of the morph-based SA heuristic, we conclude that the morphing process is a significant enhancement to SA algorithms for these problems.

Keywords: bin packing, heuristics, simulated annealing

Introduction

The classic bin-packing problem (CBPP) in one dimension can be described using notation consistent with that of Hall et al 1, as follows:

$$\text{Min } \sum_{j=1}^{n} Y_j \tag{1}$$

Subject to

$$\sum_{i=1}^{n} t_i X_{ij} \leq C Y_j \qquad j = 1, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad i = 1, \ldots, n \tag{3}$$

$$X_{ij} \in \{0,1\} \qquad i = 1, \ldots, n \text{ and } j - 1, \ldots, n \tag{4}$$

$$Y_j \in \{0,1\} \qquad j = 1, \ldots, n \tag{5}$$

Where $Y_j = 1$ if bin j is used, 0 otherwise;

$X_{ij} = 1 \; = \; 1$ if item i is assigned to bin j, 0 otherwise;

$n =$ the number of items;

$t_i =$ the size of item i; and

$C =$ the capacity of each bin ($C \geq \max_{1 \leq i \leq n}\{t_i\}$).

The objective of the CBPP is to pack n items of size $t_i (1 \leq i \leq n)$ n) into bins of fixed capacity C, while using as few bins as possible. The objective function (1) is to minimize the number of bins that are used to pack items. Constraints (2) ensure that the capacity of each bin is not exceeded. Constraints (3) require that each item is assigned to exactly one bin. Constraints (4) and (5) place binary restrictions on the items assignment and bin usage variables, respectively. It is well-known that the CBPP is NP- complete [2,3] and, therefore, a substantial research effort has been devoted to the development of heuristic procedures that can find good, but not necessarily optimal, solutions. Coffman *et al*. [4] provide an excellent survey of some of the most prominent heuristic methods for the CBPP. The worst- case performance associated with some of these

methods has been widely studied [5−7], and, to a lesser extent, average- case performance has been examined [1].

Some interesting modifications of the CBPP have also been investigated. For example, Kiimpke8 considered a problem in which the number of bins was fixed and items were packed in bins with the objective of minimizing the maximum workload assignment (across all bins). This paper focuses on a similar modification of the CBPP, in which the number of bins is known and each bin is assumed to have unlimited capacity. The objective is to assign items such that the workload (allocation) across all bins is as evenly distributed as possible. This problem, hereafter referred to as the modified bin-packing problem (MBPP), was recently described by Rao and Iyengar [9], who noted its applicability to problems such as batch processing. In their example, computer tasks of various sizes were to be assigned to a fixed number of processors, such that each processor has approximately the same workload. This type of application can be generalized to encompass many types of practical problems in which the objective is to distribute workload, as evenly as possible, across homogeneous machines, groups, or individuals.

Rao and Iyengar [9] observed that the target workload assignment, *T*, for the MBPP is $T = (\sum_{i=1}^{n} t_i)/B$, where *B* is the number of bins. Further, they proposed that an even workload across all bins should be promoted by minimizing the sum of squared deviation between T and the bin workload.

If W_j denotes the total workload in bin}, then MBPP can be formally stated as follows:

$$Min \sum_{j=1}^{B} (T - W_j)^2 \qquad (6)$$

Subject to

$$\sum_{i=1}^{n} t_i X_{ij} = W_j \qquad j = 1, \dots, B \qquad (7)$$

$$\sum_{j=1}^{B} X_{ij} = 1 \qquad i = 1, \dots, n \qquad (8)$$

$$X_{ij} \in \{0,1\} \qquad i = 1, \dots, n \text{ and } j = 1, \dots, B \qquad (9)$$

The objective function (6) of MBPP is to minimize the sum of squared deviation between *T* and W_j for $1 \leq j \leq B$. Constraints (7) are used solely for the convenience of expressing the workload of each bin as a single variable, W_j. Constraints (8) ensure that each item is assigned to exactly one bin, and constraints (9) place binary restrictions on the item assignment variables.

It is possible to obtain a lower bound (LB) for equation (6) of the MBPP. We begin by defining *p* as the precision of item sizes (for example, if the item sizes vary in integer increments, then p = 1), and ⌊x⌋ as the largest integer that does not exceed *x*. The best possible solution would have Q_1 bins of size T_{lower} and Q_2 bins of size ($T_{\text{lower}} + p$), and so,

$$LB = Q_1(T - T_{\text{lower}})^{\,2} \qquad (10)$$

Where

$$T_{\text{lower}} = p\lfloor T/p \rfloor \qquad (11)$$

$$Q_2 = (T - T_{\text{lower}})B/p \qquad (12)$$

And

$$Q_1 = B - Q_2 \qquad (13)$$

We define a 'perfect pack' to be a solution to an MBPP having an objective of LB, but note that LB (perfect packs) might not be attainable for many MBPPs.

Rao and Iyengar [9] noted that the solution space for MBPP is exponential in the size of the item list and, therefore, heuristic algorithms would be necessary for problems of practical size. Based on this premise, they developed a heuristic based on the simulated annealing (SA) algorithm for solving MBPP. Their procedure used a randomly generated starting solution and two types of random neighborhood search procedures in the SA heuristic. The first procedure randomly selected an item from a bin and moved the item to another, randomly selected bin. This

search procedure was used for the first part of the execution of the SA heuristic. The second search procedure selected two items and interchanged their bin locations. This procedure was used in the latter stages of the SA heuristic. In a computational study, Rao and Iyengar showed that their SA-based heuristic yielded better solutions than a number of greedy heuristics.

The primary limitation of the completely random neighborhood search procedure associated with Rao and Iyengar's SA based heuristic for MBPP is that it evaluates many interchanges that have little chance of improving the objective function. This is particularly true in later stages of the heuristic after a 'reasonably good' incumbent solution has been obtained. The neighborhood search procedure presented herein promotes interchanges that are more likely to improve the objective function by limiting such interchanges to similarly-sized items. By limiting potential interchanges to items which are comparable in size (which we refer to as morphs), the heuristic avoids the investigation of many poor (namely objective-function increasing) interchanges. We refer to this new method as a morph-based (as opposed to a completely random) SA heuristic.

In the next section of this paper, we present our SA-based heuristic that uses information regarding item sizes when looking for improved solutions to MBPP. In subsequent sections, we describe a large computational study developed to compare our method to that of Rao and lyengar [9], present the results of that study (which indicates that the morph-based SA heuristic substantially outperforms Rao and lyengar's random-search method), and summarize our results and offer suggestions for future research.

A morph-based simulated annealing heuristic

The SA algorithm [10−12] has been successfully applied to a variety of combinatorial optimization problems, including the quadratic assignment problem [13], the set-covering problem [14−15], manufacturing cell scheduling [16], workforce scheduling [17,18], and modified forms of the bin-packing problem [8,9]. Thompson [18] originally proposed the concept of 'morphing' as a supplement to SA-based heuristics for workforce scheduling. Thompson et al. [15] extended the approach to set-covering problems, and showed that morphing significantly increased the performance of an SA heuristic.

It is the premise of this paper that the use of morphs can also enhance the performance of SA-based heuristics for MBPP.

For the MBPP context, we define item i as a morph of item *k* if these two items are similar in size. More specifically, we construct, for each item, a morph list which contains M similarly-sized items. We assume without loss of generality that the items have been ranked in order of non-increasing sizes, namely $t_i \geq t_{i+1}$, for $1 \leq i < n$. For each item, *i*, $(1 \leq i \leq n)$, the following procedure was used to construct the morph list

$Step\ 0$ Set $k = h = i$. Set $icontinue = m = 0$.

$Step\ 1$ $k = k - 1$.

$Step\ 2$ If $(k \geq 1, t_k > t_i,$ and $m < M)$, then set $m = m + 1$,

$Step\ 3$ Set $h = h + 1$.

$Step\ 4.$ If $(h \leq n, t_h < t_i,$ and $m < M)$, then set $m = m + 1$, $B_{im} = h,$ and $icontinue = 1$.

$Step\ 5.$ If $m = M$ or $icontinue = 0,$ then **STOP**. Otherwise, set *icontinue = 0* and return to step 1.

As shown above, the morph selection process attempts to select for each item, *M/2* items with the smallest larger item sizes and *M/2* items with the largest smaller item sizes. If fewer than *M/2* morphs with larger (smaller) sizes are available, then the majority of the morph list will be comprised of items of smaller (larger) size. The final output of the morph selection process is an $n \times M$ matrix, consisting of *M* morphs for each item.

The morph list serves as the primary instrument for the search process of the SA-based heuristic. At each iteration, an item, *i* is randomly selected. Next, an item, *k*, is randomly selected from item *i*'s morph list. A check is made to ensure that item k and item i are not in the same bin. If k and i are in the same bin, then k is replaced with another item from the morph list. The change in the objective function (6) resulting from interchanging item i in bin} with item *k* in bin *m* is then determined based on the following equation [9]

$$\delta = 2(t_k - t_i) \quad (\; W_j - \quad W_m \quad - \; t_i + t_k) \\ \{j|x_{ij} = 1\} \; \{m|x_{km} = 1\} \qquad (14)$$

If δ is negative, then the objective function of MBPP (6) will be reduced and the bin locations of items *i* and *k* are switched. If δ is positive, then the probability of switching bin locations of items i and k is computed as $e^{-\delta/R}$, where *R* is the 'temperature' in the annealing analogy. Throughout the SA-heuristic, the temperature is gradually reduced by a cooling factor, *c*. There are f temperature reductions performed during the execution of the SA heuristic and, at each temperature, a temperature length of d pairwise interchanges of items are evaluated. Thus, the SA-based heuristic will evaluate up to $f * d$ solutions during its execution. If the heuristic identifies a solution with an objective value of *LB*, then it detects and returns this known optimal solution and terminates the heuristic execution.

Experimental study

The heuristic procedures

Preliminary investigation revealed that the performance of our new SA heuristic with morphing (hereafter SAWM) was sensitive to parameter choices regarding the length of the morph list (*M*), the initial temperature (R_0), and the cooling factor (c). Therefore, we constructed 12 unique versions of SAWM associated with all combinations of three morph list sizes, two initial temperatures, and two cooling factors. Although heuristic performance is also sensitive to the *d* and *f* parameters, we set these respectively at 20 000 and 250 for all SAWM versions. This results in each version of SAWM evaluating $5 * 10^6$ solutions across a broad range of temperatures, and generally is sufficient for a 'frozen' solution to be reached. For each of the 12 SAWM versions, our best results were obtained when using a first-fit decreasing (FFD) heuristic to generate initial starting solutions and all results presented herein are based on such starting solutions.

When determining the appropriate morph list size, it is desirable to have a morph list which is large enough to enable interchanges that provide reductions in the objective function, yet not so large as to result in the evaluation of many moves which cannot possibly reduce the objective function. The three morph list sizes evaluated in this study were $M = 10, M = 30,$ and $M = 60$.

The initial temperature should be specified large enough to allow a reasonable chance of accepting an objective-function increasing interchange, yet not so large as to result in too many such interchanges being accepted during the execution of the SA algorithm. Briefly, we selected initial temperatures such that 20 or 50% of the inferior, initial, 30-morph-based moves (away from the FFD solution) would be accepted. We operationalized this as follows. We first found

the FFD solution and then we randomly evaluated 1000 morph-based interchanges (without ever altering the FFD solution) and stored the change in the objective function that would result from such interchanges in a vector, δ. These interchanges were based on a morph list of 30 items. Next, we generated 1000 uniform random numbers on the interval [O, 1) and stored them in a vector, δ. The initial temperature, R_0 was subsequently determined (using a simple search procedure) such that $e^{-\delta/R_0} \leq n_i$ for a specific percentage of the objective function increasing interchanges, n. The levels defining the low and high initial temperatures, R_0, were π = 0.2 and π = 0.5, respectively.

The cooling factor should be large enough to ensure that, at the end of the execution of the SA algorithm, the probability of accepting an objective function-increasing interchange is rather small. However, the cooling factor should not be so large as to reduce the temperature too quickly, thus over-restricting the chance of accepting objective function-increasing interchanges too early in the algorithm. In this study, we evaluated cooling factors of $c = 0.95$ and $c = 0.98$. Throughout the remainder of the paper, we refer to the 12 versions of the SAWM heuristic via their defining triplet: SAWM(M,R_0, c).

We compared the 12 versions of the SAWM heuristic to two versions of the random-search SA heuristic described by Rao and Iyengar [9]. The first of these procedures (hereafter Rl-O) uses Rao and lyengar's original procedure [9], without modification. The second procedure (hereinafter RI-M) uses a modified version of Rao and Iyengar's method in which the FFD heuristic is used to obtain the initial solution. We included the modified version of Rao and Iyengar's method (RI-M) since we found that much of the improvement that SAWM was yielding compared to RI-0 was a result of SAWM's superior (FFD) starting solution. Thus, to avoid the confounding effect introduced by different starting solutions, we conclude RI-M.

Rao and Iyengar [9] specify their initial and final 'temperatures,' R_0 and R_f, respectively, as

$$R_0 - 2t_1(BT - t_1)/\log(n) \quad (16)$$

$$R_f = 2t_n^2/\log(n) \quad (17)$$

Throughout the RI-0 and RI-M heuristics, the temperature is gradually reduced by a cooling factor of c = 0.95, according to Rao and Iyengar's recommendation. We then calculate the number of temperature reductions, NREDUC, that will occur given RI-0 and RI-M's initial and final temperatures (from equations (16) and (17)) and their cooling factor

$$NREDUC = \left(\log(R_0) - \log\left(R_f\right)\right)/\log(c) \quad (18)$$

Then, to ensure that these heuristics, like SAWM, evaluate $5 * 10^6$ solutions, the temperature length, *d*, is set to $5 * 10^6$/NREDUC pairwise interchanges of items. As for SAWM, if the RI-0 (or RI-M) heuristic identifies a solution with an objective value of *LB*, then it detects and returns this known optimal solution and terminates the heuristic execution.

The test problems

The test problems for this study were generated within the context of a three-factor (3 x 6 x 3) experimental design with three replications per cell (54 x 3 = 162 total test problems). For the first factor, number of items, the three factor levels were *n* = 100, *n* = 1000, and *n* = 10,000. Thus, we investigate a much wider range of *n* then Rao and Iyengar [9], who used *n* = 1000 for their entire study. The six levels for the second factor, number of bins, were *B* = 0.05*n*, *B* = O. ln, *B* = 0.2*n*, *B* = 0.3n, B = 0.4n, and *B* = 0.5*n*. At the *B* = 0.05*n* factor level, there tends to be significant flexibility for packing items into bins, since each bin will contain an average of 20 items. At the *B* = 0.5*n* level, there is much less flexibility, since each bin contains an average of only two items. The third factor we investigated is the range of item sizes. For all three levels of

this factor, item sizes were generated based on a uniform distribution on the interval [1, *a*]. The values defining these levels were *a*= 100, *a*= 10 000, and *a*= 1 000 000.

Fourteen heuristics-the twelve SAWM versions, RI-O, and RI-M-were applied to each of the 162 test problems. All procedures were written in FORTRAN and implemented on a P5-90 personal computer.

Experimental results

An aggregate summary of the experimental results

An aggregate summary of the results of the experimental study is presented in Table 1. The five performance measures in Table 1 include, for each heuristic: (1) the number of test problems (out of 162) for which the procedure identified the best known objective value (namely, the best objective function value across all heuristic procedures); (2) the number of test problems for which the procedure identified perfect packs (known optimal solutions of *LB*); (3) a comparison of its solutions to those generated by RI-M; (4) the mean percent above the best known objective values (MPABKOV); and (5) the average number of CPU seconds required to find its best heuristic solution.

We begin our aggregate comparison by noting that RI-M was, on average, superior to RI-O in terms of all performance measures. This supports our earlier contention regarding the importance of a good starting solution. To avoid confounding the effect of different starting solutions with the effect of how item interchanges are selected, our subsequent analysis will therefore utilize only RI-M for comparative purposes.

It is clearly evident that all 12 SAWM versions substantially outperformed Rl-M in terms of all performance measures, with the exception of CPU time required to find the best solution.

While RI-M identified its best heuristic solution about twice as quickly as the SAWM versions, the significance of this finding is mitigated by the fact that RI-M was generally not finding the best heuristic solution. That is, RI-M was more efficient than SAWM at finding the best solution it was capable of obtaining; however, RI-M's solution was typically much worse than the solutions yielded by SAWM.

With respect to MPABKOV, the four best SAWM procedures were ranked as follows: SAWM(10,High,0.95), SAWM(30,High,0.95), SAWM(10,Low,0.95), and SAWM(60,High,0.95). Thus a high initial temperature in conjunction with a fast cooling rate seemed to be most conducive to superior performance on this criteria. The SAWM procedures with the worst MPABKOV were associated with high initial temperatures and slow cooling rates.

SAWM(30,High,0.95) identified 66 best known objective values, three more than its nearest competitors, SAWM(30,Low,0.95) and SAWM(60,High,0.95). Once again, a high initial temperature and fast cooling rate seemed to yield the best performance in terms of this criterion, with a fast cooling rate being especially important. It should be noted that procedures associated with the larger morph list sizes tended to yield more best known solutions than procedures for which M = 10.

SAWM(30,Low,0.95) identified the most known optimal solutions (49), one more than SAWM(60,High,0.95), two more than SAWM(30,High,0.95), and three more than SAWM(60,Low,0.95). Clearly, the faster cooling rate was more important for superior performance on this criterion and was, once again, best supported by a high initial temperature.

The SAWM (10 ,Low, 0.95), SAWM(10, High, 0.95), and SA WM(30,High,0.95) procedures each generated objective values that were less than or equal to (i.e., better than or the

same as) those of RI-M for 148 or the 162 test problems. Of these three procedures, SAWM(30, High, 0.95) was superior, yielding objective values that were strictly less than those of RI-M for 127 test problems. Even the worst SAWM procedure, SAWM(60, High, 0.98), yielded lower objective values than RI-M for 97 test problems, and higher objective values than RI-M for only 45 test problems.

There is clearly some disparity in the relative performance of the SAWM procedures across different criteria. However, we view MPABKOV as the most important performance criterion, since it provides the best average measure of the quality of heuristic solutions. Thus, given its superior MPABKOV performance, we will use SAWM(10, High, 0.95) for our subsequent comparison to RI-M. Relative to RI-M, SAWM(10,High,0.95) provided 130% more best known objective values and 50% more known optimal solutions. In addition, SAWM(10, High, 0.95)s objective values were better (that is less) than those of RI-M for 75.3% of the test problems, equal to those of RI-M for 16.1% of the test problems, and worse (that is greater) than those of RI-M for only 8.6% of the test problems. Perhaps the greatest evidence of the superiority of SAWM (10, High, 0.95) is provided by the fact that its average departure from the best known objective value was only 56.868%, while the corresponding figure for RI-M was 16 534.05%. These results suggest that RI-M was not only inferior for most problems, but could provide very poor solutions for some problems as well.

A detailed summary of the experimental results

Table 2 provides a MPABKOV comparison between RI-M and SAWM (10, High, 0.95) across the different levels of number of items, item size range, and number of bins. It is evident

that the SAWM procedure provided a much lower MPABKOV than RI-M at all levels of each of these environmental factors.

SAWM's relative superiority procedure increased with more items, a wider range of item sizes, and when the number of bins was small relative to the number of items. RI-Ms MPABKOV values were 16.6, 253.2, and 564.9 times larger than those of SAWM (10, High, 0.95) for the 100, 1000, and 10 000 item test problems, respectively. RI-Ms MPABKOV values were 2.3, 150.8, and 825.5 times larger than those of SAWM (10, High, 0.95) for the a = 100, a = 1000, and a = 10 000 item test problems, respectively. Finally, RI-Ms MPABKOV values were 766.6, 930.4, 303.9, 10.3, 24.9, and 14.2 times larger than those of SAWM(10, High, 0.95) for the $B = 0.05n$, B + 0.10n, B = 0.20n, $B + 0.40n$, and $B + 0.50n$, test problems, respectively.

The results in Table 2 pertaining to the number of items and item size distribution were particularly supportive of our expectations regarding the types of MBPPs for which the morphing process would be most effective. For example, RI-M and SAWM yield their most similar performance in MBPPs with fewer items. This is attributable to the fact that SAWM uses a fixed morph list size, and proportionally more of the total items are on any morph list when the total number of items is smaller (which results in more similarity between totally random and morph-based interchanges). As n increases, a smaller proportion of the total items are on any morph list and the benefits of the morphing process becomes more pronounced.

The relative performance difference between RI-M and SAWM also tends to be smaller for problems with less dispersion in the item sizes. As the item size dispersion increases (i.e., namely the parameter a increases), two items selected totally at random could have substantially different sizes. This makes totally random interchanges perform less similarly (and less

effectively) compared to morph-based interchanges at high item size dispersion than at low item size dispersion.

Using multiple SAWM procedures

Table 3 provides an examination of the reduction in MPABKOV associated with using multiple versions of SAWM. The idea behind this is to consider the benefit of running multiple versions of SAWM on each problem and selecting the best of the solutions found. From Table 1 we know that the single best version of SAWM was SAWM (10, High, 0.95), which had a MPABKOV of 56.868. Should another version be run, it is SAWM (30, High, 0.95). Then if the best solution of these two versions is selected, MBABKOV drops markedly from 56.868 to 21.450. These two procedures would then be best supplemented by SAWM (10, Low, 0.95), reducing MPABKOV by 34% to 14.171. If a fourth procedure were added, the best such procedure would be SAWM(60,Low,0.98) and MPABKOV would be reduced to 9.361. Reductions in MPABKOV begin to diminish rapidly when five or more procedures are examined.

Conclusions

Summary of experimental results

The computational results presented herein showed that SAWM significantly outperformed RI-0 and RI-M across a variety of test-problem characteristics and performance measures. The superiority of SAWM is attributable to its use of a morph-based search process during the execution of the SA heuristic. The search phase of the RI-0 and RI-M heuristics selects two items at random, and evaluates the effect on the objective value (6) associated with

switching their bin locations. Unfortunately, many of these interchanges will have very little chance of improving (6), particularly during the latter stages of heuristic execution. For example, consider the case for which a= 10 000. Suppose that the two randomly selected items, i and k, have item sizes of t;= 500 and tk= 9500, respectively. It is very unlikely, particularly in later stages of the heuristic, that switching items i and k could have a beneficial effect on (6). However, many potential interchanges such as this one will be evaluated in the RI-0 and RI-M heuristics. The search phase of SAWM selected the first item at random from the set of all items. However, the second item is randomly selected from the first item's morph list. Thus the two items for which an interchange will be evaluated will have comparable t; values, improving the chances of a beneficial interchange.

There was no single version of the SA WM procedure that particularly dominated the other procedures in terms of solution quality. However, there was a notable difference in performance associated with the different cooling factors. That is, procedures that used a faster cooling rate (c = 0.95) tended to provide more best known solutions, more known optimal solutions, and better MPABKOV values than those using the slower cooling rate (c = 0.98). Obviously, these findings are dependent on our choices of d and f.

There is considerable evidence to suggest that running multiple versions (namely, different parameter settings) of the SAWM heuristic can lead to substantial improvement in solution quality. Supplementing the best procedure, SAWM(10, High, 0.95), with just one other procedure, SAWM(30,High,0.95), resulted in a 62% decrease in MPABKOV.

For the test problems in this study, optimal solution were only detected if the objective function obtained by a heuristic procedure was equal to LB. The SAWM procedures typically generated known-optimal solutions for 20-30% of the test problems. However, it is conceivable

that many more optimal solutions were obtained, but not identified as optimal, since the optimal objective function value was greater than *LB*.

Research extensions

There are at least two major avenues of research that can stem from this paper. First, it should be possible to identify enhancements to the morph-based SA procedure. In this study, we made efficient use of the morph list concept by randomly selecting item *i* and then selecting one of its morphs at random. A more comprehensive (though computationally more burdensome) search could be conducted by examining all of item i's morphs, and the effect of their interchange with item *i*. The interchange providing the greatest reduction of (6) would be selected. Another possible modification would be to bias the probability of selecting item *i* from a bin which makes a substantial contribution to (6). A third possible enhancement of SAWM would enable it to alter the number of items in bins. Since SAWM only swaps items between bins, it never alters the initial number of items in bins. Clearly, this may hinder its ability to find good solutions on some problems. However, SAWM's superior performance indicates that any improvements so achieved may be modest, particularly compared to the performance of RI-0 and RI-M which can alter the number of items in bins early in their execution. Finally, the results reported in Table 2 with respect to item size dispersion suggest that it may be appropriate to vary the length of the morph list depending on problem characteristics, selecting more morphs when the items are more similarly sized and fewer morphs when the items are more dissimilarly sized.

Second, it would be interesting to extend the search concepts we developed for the MBPP to the CBPP. This extension is complicated by the fact that the CBPP is characterized by capacity constraints which affect the feasibility of a solution. For the MBPP, as long as all items

are assigned to bins, the solution is feasible. This is clearly not the case for the CBPP. We are currently attempting to adapt the morph-based SA heuristic for the CBPP.

References

1. Hall NG *et al* (1988). Bin packing problems in one dimension: Heuristic solutions and confidence intervals *Comp Opns Res* **15**: 171-177.
2. Karp M (1972). Reducibility among combinatorial problems. ln: Miller RE and Thatcher JW (eds). *Complexity of Computer Computations*. Plenum Press: New York.
3. Garey MR and Johnson DS (1979). C*omputers and Intractability: A Guide to the Theory of NP Completeness*. Freeman: San Francisco.
4. Coffinan Jr EG, Garey MR and Johnson DS (1984). Approximation algorithm for binpacking -an updated survey. ln: Ausiello WG, Lucertini WG and Serafini P (eds). *Algorithm Design for Computer Systems Design*. Springer-Verlag: The Netherlands, pp 49-106.
5. Johnson DS et al (1974). Worst-case performance bounds for simple one dimensional packing algorithms *Siam J Comput* **3**:299-325.
6. Ong HL, Magazine MJ, Wee TS (1984). Probabilistic analysis of bin packing heuristics *Opns Res* **32**: 983-998.
7. Anily S, Bramel J and Simchi-Levi D (1984). Worst-case analysis of heuristics for the bin packing problem with general cost structures *Opns Res* **42**: 287-298.
8. T. Kampke (1988). Simulated annealing: use of a new tool in bin packing *Ann Opns Res* **16**: 327-332.
9. Rao RL and Iyengar SS (1994). Bin-packing by simulated annealing *Computs Math Applic* **27**: 71-82.
10. Metropolis Net al (1953). Equation of state calculations by fast computing machines *J Chem Phys* **21:** 1087-1092.

11. Kirkpatrick S, Gellatt Jr CD and Vecchi MP (1983). *Optimization by simulated annealing Science* **220**: 671--683.
12. Cerny V (1985). Thermodynamical approach to the travelling salesman problem *J Optim Theory Appl* **45**: 41-51.
13. Cheh K, Goldbert D and Askin R (1991). A note on the effect of neighborhood structure in simulated annealing *Comp Opns Res* **18**: 537-547.
14. Jacobs LW and Brusco MJ (1995). Note: A local-search heuristic for large set-covering problems *Naval Res Log* **42**: 1129-1140.
15. Thompson GM, Brusco MJ, Jacobs LW (1995). A metaheuristic for coverage-correlated, minimum cardinality set-covering problems. Working paper, Cornell University: Ithaca, New York.
16. Vakharia AJ and Chang YL (1990). A simulated annealing approach to scheduling a manufacturing cell *Naval Log* **37**: 559-577.
17. Brusco MJ and Jacobs LW (1993). A simulated annealing approach to the solution of flexible labour scheduling problems *J Op/ Res Soc* **44**: 1191-1200.
18. Thompson GM (1996). A simulated-annealing heuristic for shift scheduling using non-continuously available employees *Comp Opns Res* **23**: 275-288.

Table 1. Summary of results[a]

	Number of solutions						
Heuristics	*Best*	*KO*	*<Rl-M*	*=Rl-M*	*>Rl-M*	*MPABKOV*	*Time*
RI-0	22	18	39	19	104	32825.450	80.81
RI-M	26	22	0	162	0	16534.050	33.58
SAWM(lO,Low,0.95)	61	34	122	26	14	61.121	55.35
SAWM(lO,Low,0.98)	41	27	106	24	32	633.612	68.90
SAWM(lO,High,0.95)	60	33	122	26	14	56.868	58.38
SAWM(lO,High,0.98)	32	25	94	24	44	2534.719	69.76
SAWM(30,Low,0.95)	63	49	125	22	15	130.735	56.43
SAWM(30,Low,0.98)	56	41	110	22	30	762.306	66.47
SAWM(30,High,0.95)	66	47	127	21	14	60.969	60.25
SAWM(30,High,0.98)	52	38	98	21	43	2694.064	67.99
SAWM(60,Low,0.95)	61	46	123	22	17	99.333	58.75
SAWM(60,Low,0.98)	48	41	106	22	34	849.224	69.43
SAWM(60,High,0.95)	63	48	124	23	15	67.538	62.44
SAWM(60,High,0.98)	46	37	97	20	45	3198.794	68.12

[a] Best is the number of overall best solutions (of a maximum 162).

KO is the number of known optimal solutions (that is perfect packs).

< RJ-M is the number of problems (of 162) on which the heuristic yielded a better solution than RI-M.

= RJ-M is the number of problems (of 162) on which the heuristic yielded the same solution as RI-M.

> RJ-M is the number of problems (of 162) on which the heuristic yielded a worse solution than RI-M.

MPABKOVis the average, across the 162 problems, of the percent by which the heuristic's solution for a problem exceeded the best known solution (the best found by any heuristic) for the problem.

Time is the time (in seconds on a P5-90 computer) required by the heuristic to identify its best solution on a problem, averaged for all 162 problems.

Table 2. Performance comparison of RI-M and the best SAWM procedure, by problem category[a]

Problem category	*RI-M*	*SAWM(10,High,0.95)*
100 Items	1276.6	77.0
1000 Items	3697.5	14.6
10000 Items	44628.1	79.0
Size Range = 1-100	175.6	76.5
Size Range = 1-10000	6333.7	42.0
Size Range = 1-1000 000	43092.8	52.2
Bins = 0.05* Items	48834.1	63.7
Bins = 0.10* Items	32192.6	34.6
Bins = 0.20* Items	15319.0	50.4
Bins = 0.30* Items	1238.4	120.2
Bins = 0.40* Items	1382.1	55.4
Bins = 0.50* Items	238.1	16.8

[a]Performance measures are average, across the problems in each category, of the percent by which the heuristic's solution for a problem exceeded the best known objective value (the best found across all heuristic procedures) for the problem.

Table 3. Value of running multiple SAWM procedures

Number of procedures run	*Procedures*	*MPABKOV*[a]
1	SAWM(10,High,0.95)	56.068
2	+SAWM(30,High,0.95)	21.450
3	+SAWM(10,Low,0.95)	14.171
4	+SAWM(60,Low,0.98)	9.36
5	+SAWM(10,Low,0.98)	7.934

[a] Performance measures are average, across all test problems, of the percent by which the best solution of the heuristic procedures exceeded the best objective value (the best found across all procedures) for the problem.